SensorMax

App zur Erfassung mobiler Sensorik

Besondere Lernleistung

im

Fach Physik

Max – Steenbeck – Gymnasium

D–03046 Cottbus

Universitätsstraße 18

Verfasser:

Hübscher, Leonardo
Abiturjahrgang 2015
Sielower Straße 50
D-03044 Cottbus
leonardo.huebscher.business@gmail.com

Betreuer:

Hr. M. König

COTTBUS, DEN 27.03.2014

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

ABKÜRZUNGSVERZEICHNIS

Abkürzung	Bedeutung
MVC	engl.: model-view-controller
XML	engl.: Extensible Markup Language
UI	engl.: user interface
NFC	engl.: near field communication
SD	engl.: secure digital
SDK	engl.: Software Development Kit
ADT	engl.: Android Development Tools
PHP	engl.: hypertext preprocessor
CSV	engl.: character-separated values
AP	engl.: access point

1 DIE EINLEITUNG

Die Entwicklung der modernen Smartphones beeinflusst zunehmend den Alltag eines jedes Einzelnen. Heutzutage besitzen immer mehr Schüler Smartphones als Kommunikationsmittel. Sie können jedoch nicht nur für das Besuchen von sozialen Netzwerken oder zum Telefonieren genutzt werden, sondern halten auch eine Vielzahl an Sensoren für den Anwender bereit. Aus diesem Grund habe ich mich in einer Jugend-Forscht-Arbeit, welche ich mit zwei weiteren Teamkollegen erarbeitet habe, mit der Nutzung und den didaktischen Vorteilen des Smartphones im schulischen Alltag beschäftigt.

Am letzten Schultag der 11. Jahrgangsstufe führten wir eine Schülerumfrage durch. In dieser wollten wir untersuchen, wie hoch die Bereitschaft wäre, das Smartphones in den Unterricht einzubinden. Leider haben einige Mitschüler unvollständige Angaben eingereicht, sodass deren Befragungen nicht gewertet werden konnten.
Die Auswertung von ca. 160 gültigen Umfragen ergab, dass sich die Lernenden mehr Experimente im Schulunterricht wünschen. Fast ¾ der Befragten können sich vorstellen, mit dem Smartphone experimentell im Schulunterricht zu arbeiten. Besonders in den Jahrgangstufen der Sekundarstufe I war der Zuspruch recht hoch.

Deshalb haben wir beschlossen, einige Experimente mit dem Smartphone durchzuführen, um dessen Eignung für schulische Physikexperimente in Bezug auf die Messgenauigkeit festzustellen. In der Zusammenarbeit mit unseren Physiklehrer untersuchten wir im Rahmen des „Anderen Leistungsnachweises“ diesen Sachverhalt. Für die Validierung der Messdaten haben wir die mit dem Smartphones erfassten Werte mit denen des konventionellen Messsystem Cassy verglichen. Wir fanden heraus, dass die Sensorik des Smartphones für die experimentellen Zwecke akzeptable Messwerte lieferten.

Ziel dieser Arbeit ist es, eine App zu entwickeln, mit der es möglich ist, alle Sensoren des Smartphones auszulesen. Die erfassten Messreihen sollen anschließend sinnvoll visualisiert und zur Weiterverarbeitung am PC exportiert werden. Des Weiteren ist aufgrund der raschen Entwicklung in der Smartphone-Branche die Möglichkeit der Erweiterung der Applikation um neue Funktionen erforderlich. Aufgrund des didaktischen Bezuges der App ist deren Design intuitiv und übersichtlich zu gestalten.
Abschließend wird die App um die Echtzeitdatenübermittlung erweitert, sowie ein geeigneter Serverentwurf umgesetzt werden.

2 DIE ZIELSTELLUNG

Im Zuge dieser Arbeit werde ich eine App programmieren, mit der es möglich sein soll, alle Sensoren des Smartphones auszulesen. Über das Bluetooth-Kommunikationsmodul sollte es des Weiteren möglich sein, die Messwerte externer Sensoren zu empfangen und abzuspeichern. Diese Möglichkeit sollte anhand eines Beispiels umgesetzt werden. Ebenfalls werden für die Messung hilfreiche Funktionen integriert werden. Dabei richte ich mich zum einen nach den Funktionen der Konkurrenzprodukte[1] und zum anderen nach den Wünschen meiner Lehrer und Mitschüler, da diese praxisnah agieren.
Anschließend werde ich auf die aufgetretenen Probleme eingehen und erläutern, wie man eine App in den App-Store lädt.
Wenn die grundlegende Entwicklung der App abgeschlossen ist, werde ich mich mit der Einrichtung eines Servers beschäftigen, mit Hilfe dessen eine Datenübermittlung in Echtzeit ermöglicht werden soll. Dazu werde ich ein geeignetes Konzept entwerfen und umsetzten.
Abschließend werde ich die Durchführung eines Beispiel-Experimentes erörtern und auf weitere mögliche Experimente mit dem Smartphone, sowie auf mögliche zukünftige Funktionen hinweisen.

3 DIE ENTWICKLUNG DER APP

3.1 DER ENTWURF

3.1.1 DIE SENSORIK

In diesem Teil gehe ich auf die momentan verfügbaren Handysensoren ein. Dabei ist jedoch zu beachten, dass es eine Vielzahl an verschiedenen Sensoren gibt, welche Differenzen in der Qualität der erfassten Messungen aufweisen.
Ich beziehe mich daher im Folgenden auf das in der Kooperation mit Google entstandene von LG produzierte Nexus 5 als Referenzmodell.

Das Nexus 5 verfügt über folgende Sensoren: (Google, kein Datum)

- GPS
- Gyroskop
- Beschleunigungsmesser
- Magnetfeldsensor
- Barometer
- Näherungssensor
- Umgebungslichtsensor

[1] SensorKinetics und AndroSensor

Von den oben genannten Sensoren ist der Näherungssensor nicht für qualitative Messungen geeignet, da dieser lediglich zwischen einer Distanz von 0 cm und 5 cm unterscheidet.
Die allgemeine Nützlichkeit der anderen Sensoren hängt von dem durchgeführten Experiment ab und variiert je nach verwendeten Smartphone bzw. Sensor. Der Messbereich und die Messauflösung sind jedoch erfassbar, sodass der Experimentierende bereits im Voraus die zu erwartende Brauchbarkeit der Messwerte abschätzen kann.

3.1.2 DIE ANFORDERUNGEN

Nach einer gründlichen Analyse der Wünsche meiner Mitschüler und Lehrer ergaben sich folgende Anforderungen.

3.1.2.1 Das Aufnahmeverfahren einer Messreihe

Für die Messaufnahme wird die Möglichkeit der Erfassung beliebig vieler Sensoren gewünscht. Die Messungen werden dabei synchron gestartet und pausiert. Des Weiteren wäre es wünschenswert den Messintervall, die Messzeit sowie einen Messauslöser konfigurieren zu können.
Um die anschließende Auswertung zu beschleunigen sollte es möglich sein, während einer Messung Messpunkte in einer geeigneten Weise zu markieren.
Für die Ermöglichung von Langzeitmessungen wäre die Fortsetzung des Messvorgangs auch nach der Minimierung der App erforderlich.

3.1.2.2 Die visuelle Auswertung auf dem mobilen Endgerät

Für die schnelle visuelle Betrachtung der erfassten Messwerte müssen diese in einer geeigneten Darstellung (z.B. Diagramm, Spektrum, Tabelle) visualisiert werden.
Das Diagramm wird nach Möglichkeit skalierbar und scrollbar sein. Zudem sollte es möglich sein, das Diagramm in einem Vollbildmodus zu betrachten, sowie ungewünschte Messgrößen auszublenden. Um dem naturwissenschaftlichen Anspruch gerecht zu werden, ist eine Beschriftung der Achsen unabdingbar.
Neben der Darstellung der aktuellen Messgrößen wird die Anzeige weiterer messrelevanter Informationen, wie zum Beispiel die Messzeit oder auch eine Zusammenfassung einer Messung, welche die Minima, Maxima und Durchschnittswerte beinhaltet, gewünscht.

3.1.2.3 Die Weiterverarbeitung der Datensätze auf anderen Medien

Um die aufgenommen Messungen auswerten zu können, sollte es die Möglichkeit geben, diese in einem Excel-konformen Format zu exportieren. Diese Dateien sollten anschließend via Skype, NFC oder auch per E-Mail versandt werden. Außerdem soll man diese lokal auf der SD-Karte speichern können. Zusätzlich wird die Möglichkeit bestehen, die Messungen und Zusammenfassungen mit den Durchschnittswerten an einen Server zu senden.

3.1.2.4 Die weiteren Funktionen

Über die normalen Funktionen hinaus, werden aus dem physikalischen Kontext heraus, ein Frequenz-Ton-Generator, sowie eine Audioanalyse gewünscht.
Das Empfangen von Messwerten externer Sensoren wird anhand des Mindwave-Think-Gear-Geräts demonstriert werden.

Ebenso sollte es möglich sein, die Displayrotation zu deaktivieren und das Abschalten des Bildschirms während einer Messung zu verhindern.

3.1.2.5 Die didaktischen Anforderungen

Für den erfolgreichen Einsatz der App im Unterricht ist nicht nur die korrekte Funktionsweise zu garantieren, sondern auch dem Anwender ein ansprechendes Design zu präsentieren.

3.1.3 DIE SOFTWARE-ARCHITEKTUR

Für den Grobentwurf der Applikation werde ich mich an das „Model-View-Controller"-Konzept halten. Es sieht eine Aufteilung in drei Einheiten vor: dem Datenmodell, der Präsentation und der Programmsteuerung. Ich habe mich für diesen Entwurf entschieden, da dieser besonders für spätere Erweiterungen geeignet sein soll. Ebenfalls gilt das MVC-Konzept als Standard für viele Softwaresysteme.
(Unbekannt, Model View Controller, 2015)

Das Betriebssystem Android erleichtert dem Programmierer sich an dieses Konzept zu halten, indem die Trennung der Oberfläche durch das Auslagern der UI in XML-Dateien vorgegeben wird. Die Aufteilung von Modell und Steuerung hat der Programmierer jedoch selbst beachten.

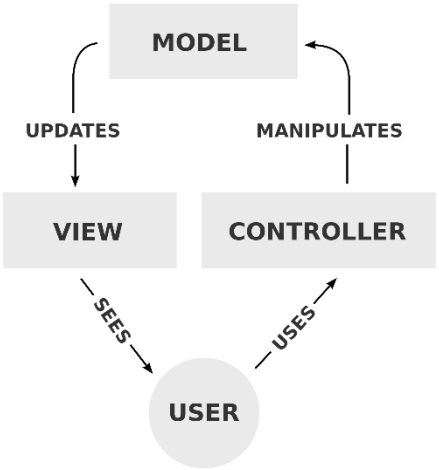


Abbildung 1 Model-View-Controller-Konzept

Quelle:http://upload.wikimedia.org/wikipedia/commons/thumb/a/a0/MVC-Process.svg/2000px-MVC-Process.svg.png

3.1.4 Die Hierarchie der Klassen

In diesem Teil der Arbeit werde ich auf den allgemeinen Aufbau des Programms eingehen, um das Verständnis für die technische Funktionsweise zu steigern.

Zuerst wird die App über den Android-Launcher gestartet. Dadurch wird die *DataHandlerActivity* aufgerufen. Sie erbt von der *NavigationActivity*, welche sich um die Verarbeitung des linken Slide-In-Menüs kümmert. Somit kann das Menü, welches man durch Wischen von links nach rechts öffnen kann, in der App verwendet werden. Die *NavigationActivity* wurde von der Programmierumgebung Eclipse generiert.
Um den Informationsaustausch innerhalb der App zu vereinfachen, habe ich alle anderen relevanten Klassen in der *DataHandlerActivity*-Klasse referenziert.

Grundsätzlich gibt es die drei Pakete: *Model*, *View* und *Controller*. In dem Paket *Controller* befinden sich alle *Fragments*. Diese weisen den UI-Elementen eine Funktion zu und dienen somit als Bindeglied zwischen Anzeige und Modell.
In dem *Model*-Paket befinden sich alle funktionale Klassen, wie z.B. der *LivestreamManager*, welcher sich um das Senden der Messwerte an den Server kümmert. Die Klasse *MyConfig* lädt alle Programmparameter aus dem App-Speicher, wie z.B. die gespeicherten Einstellungen.
Des Weiteren gibt es eine Klasse *Measurement*, welche die grundsätzlichen Funktionen der Messung enthält. Diese Klasse wird durch erbende Klassen spezifiziert. Zusätzlich gibt es eine Klasse für jeweils einen Sensor. Alle erben von der Klasse *MySensor* und geben die ausgelesenen Sensorinformationen an diese weiter.

Eine ausführliche Auflistung ist dem vereinfachten Klassendiagramm (Anhang 1) zu entnehmen.

3.2 Die Umsetzung

3.2.1 Die Einrichtung der Programmierumgebung

Für das Programmieren habe ich die Programmierumgebung Eclipse verwendet.
Um Eclipse einzurichten sind folgende Schritte zu befolgen:

1. Eclipse herunterladen:
   https://eclipse.org/downloads/packages/eclipse-ide-java-developers/lunasr1
   und am besten nach C:/Program Files/Eclipse\eclipse verschieben
2. Android-SDK herunterladen und nach C:\Program Files\Eclipse\sdk installieren:
   https://developer.android.com/sdk/index.html#Other
3. Java-SDK herunterladen und nach C:\Program Files\Java installieren
   http://www.chip.de/downloads/Java-SDK-64-Bit_47299013.html
4. Eclipse öffnen → Hilfe → Neue Software installieren
   a. ADT - https://dl-ssl.google.com/android/eclipse

   b. Babel Language Pack - http://download.eclipse.org/technology/babel/update-site/R0.12.0/luna (Dieses stellt eine deutsche Oberfläche für Eclipse zur Verfügung)
   c. GitHub-Plugin - http://download.eclipse.org/egit/updates
5. Git-Perspektive in Eclipse öffnen und dieses[2] Git-Repository klonen.

Beim Anschließen des Handys sollte darauf geachtet werden, dass der Zugriff des Computers auf das Handy genehmigt werden muss. Um die App auf dem Handy zu installieren führt man das Programm durch einen Klick auf den grünen Button in der Menüleiste aus. Möglicherweise muss vorher das Installieren von Apps unbekannter Herkunft genehmigt werden. Dazu muss man in die Sicherheitseinstellungen des Smartphones gehen und das Installieren von Apps aus anderen Quellen erlauben.

3.2.2 Die Erfassung der Messdaten der Sensorik

Für die Erfassung der Sensorik stellt die Android-Bibliothek bereits eine eigens dafür entwickelte Software-Bibliothek zur Verfügung. Damit der Applikation die aktuellen Werte mittgeteilt werden, wird zuerst ein Listener im System registriert. Das System prüft nun, ob sich der erfasste Wert des Sensors geändert hat und sendet gegebenfalls den neuen Messwert an den Listener. Der Listener aktualisiert daraufhin die Informationen in der App. Ist der Wert unverändert geblieben, so passiert nichts. Wenn die Applikation keine weiteren Daten eines Sensors erhalten soll, wird abschließend der Listener abgemeldet. (Android, kein Datum)

3.2.3 Die Audioanalyse

Die Audioanalyse dient der Erfassung der relativen Lautstärke aller Frequenzen. Dazu wird zunächst auf das aufgenommene Audiomaterial eine Fourier-Transformation ausgeführt. Diese zerlegt die momentan zu hörende Grundfrequenz in ihre einzelnen Frequenzen. Sie ist somit die Umkehrung der Überlagerung von Audiowellen. Anschließend kann anhand dieser Informationen ein visuelles Spektrum angelegt, sowie die dominanteste Frequenz bestimmt werden. Durch ein Näherungsverfahren kann zusätzlich versucht werden, die aktuelle Lautstärke in Dezibel zu ermitteln. Für die Fourier-Transformation verwende ich die Audio-Analysis-Bibliothek[3]. (Madusanka, 2012) (Ruge, 2013)

3.2.4 Die Echtzeit-Übermittlung der Messwerte

Die Schwierigkeit dieser Umsetzung lag vor allem darin, den Verlust von Messwerten zu minimieren und dennoch eine hohe Übermittlungsgeschwindigkeit zu erreichen. Dies wurde erreicht, indem alle zu sendenden Daten als erstes in eine Schlange kopiert wurden. Parallel zu den neu anfallenden Daten werden die Messwerte der *Schlange* nacheinander an den Server gesendet. Erst wenn dieser die erfolgreiche Ankunft der Daten

[2] https://github.com/deeps96/SensorMax
[3] www.netlib.org/fftpack/jfftpack.tgz

bestätigt, wird der gesendete Messwert aus der Schlange gelöscht. Sollte die Internetgeschwindigkeit allerdings zu gering sein, so könnte dies zu Problemen führen, da der RAM-Speicher des Smartphones überlaufen könnte. Da man sich jedoch meistens während einer Messung in einem gemeinsamen lokalen Netzwerk befindet, sollte dies kein ernsthaftes Problem darstellen.

3.2.5 Der Ton-Frequenz-Generator

Diese Umsetzung lässt sich mit einfachen Mitteln erreichen. Am Anfang wird ein Array mit den Werten der Sinus-Kurve einer gewünschten Frequenz gefüllt. Danach wird das Array an ein Audio-Track-Objekt der Android-Bibliothek übergeben, welches die Daten verarbeiten kann und den Ton abspielt. Um das Klick-Geräusch, welches zwischen den Datenblöcken auftritt zu unterbinden, schwillt die Lautstärke am Anfang jedes Blockes an und am Ende ab. (Xarph, 2012)

3.2.6 Die Erfassung der Daten des Think-Gear-Zubehörs

Das Think-Gear-Headset von Mindwave dient der Erfassung von Hirnströmen (EEG). Die Daten werden von dem Headset per Bluetooth an das Gerät versendet.
Um diese übermittelten Informationen interpretieren zu können, wird die vom Hersteller Mindwave zur Verfügung gestellte Bibliothek benötigt. Diese erleichtert einem die Auswertung, sodass die Daten der Kanäle 1 – 8, sowie der Herzschlag einfach ausgelesen und in einer Tabelle ausgegeben werden können. Zusätzlich liefert die Bibliothek Methoden zur Erfassung des aktuellen Status des Gerätes, wie zum Beispiel ob es momentan verbunden ist, oder ob es noch genügend Akkuladung besitzt. Diese Status-Meldungen werden über Einblendungen mitgeteilt.

3.2.7 Die Funktionsweise der Messaufnahme

Dieser Abschnitt beschreibt die Funktionsweise einer Messaufnahme anhand eines Quellcodeausschnitts der *Measurement*-Klasse.
Als erstes werden die benötigten Variablen in der Methode *initialise()* in der Klasse *Measurement* initialisiert und Einstellungen aus dem App-Speicher gelesen.
Die Variable *maxDataCounter* gibt an, wie viele Messreihen je Messung im RAM des Gerätes gehalten werden, bis die Daten überschrieben werden.
Das Array *triggerValues* enthält die Werte für die Messauslösung, also den Wert, den eine bestimmte Messgröße haben muss, ehe die Messung gestartet wird. Dieser Wert wird mit der minimalsten Größe des Datentyps Float initialisiert und kann durch ein Fragment manipuliert werden.

In der *run*-Methode steht der Quellcode, welcher in einem parallelen Thread ausgeführt wird, damit nicht die grafische Oberfläche blockiert wird. In diesem werden zunächst die erforderlichen Messwerte mit jedem Start einer Messung initialisiert.

Die Methode *registerListener*, welche vor der while-Schleife aufgerufen wird, teilt den Sensoren mit, dass nun Werte angefordert werden.
In der while-Schleife werden im Zeitabstand des Messintervalls die aktuellen Werte, welche im *recentDataSet*-Array gehalten werden, in das *data*-Array gespeichert. Die Werte des *recentDataSet*-Array werden durch eine update-Methode in der erbenden Klasse verändert. Ebenfalls werden die aktuellen Werte an weitere Untermethoden weitergegeben. Eine davon prüft, ob der Mess-Trigger ausgelöst wurde. Eine andere berechnet wiederrum die Durchschnittswerte. Abschließend werden die Werte an das Element übergeben, welches die Messreihe visualisiert.
Wenn die Messung an einen Server gesendet werden soll, werden der entsprechenden Methode im *LivestreamManager* die Daten übergeben.

3.2.8 Die Screenshots

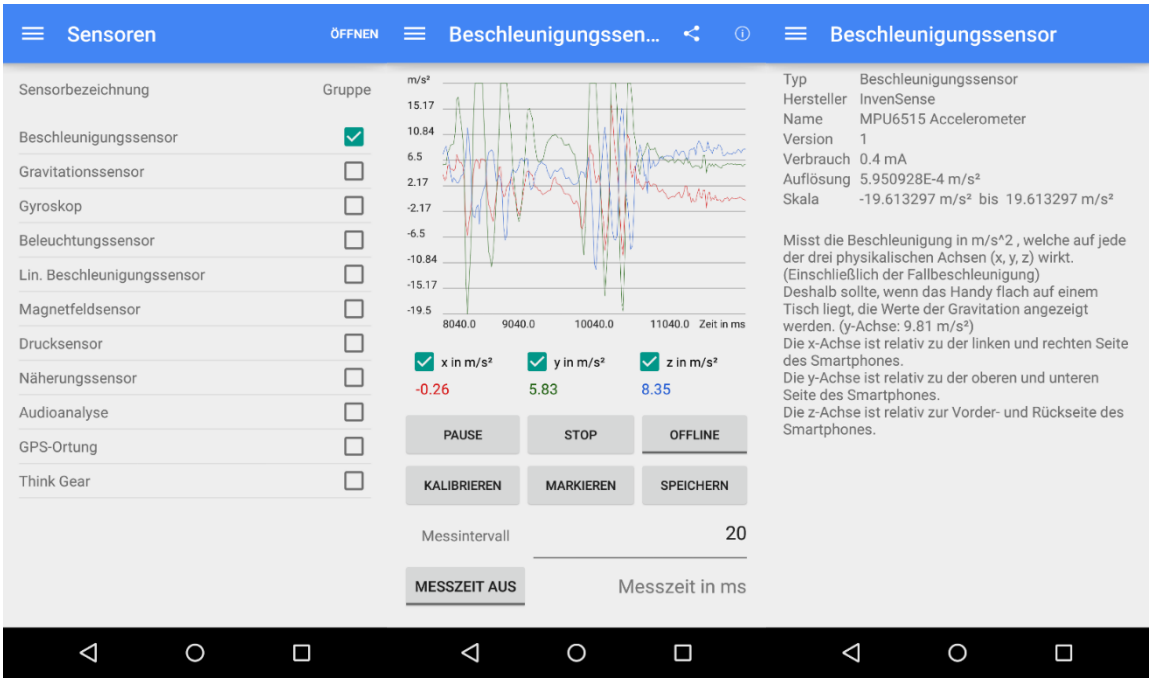


Abbildung 2 Screenshots der App SensorMax

3.2.9 Die Veröffentlichung der App bei GooglePlay

Um eine Installationsdatei in die Produktionsphase auf GooglePlay hochzuladen sind lediglich wenige Schritte zu beachten. Zu allererst sollte man die Versionsnummer in der Manifest-Datei erhöhen. Es empfiehlt sich ebenfalls den Versionsnamen zu ändern. Anschließend klickt man in dem Paketexplorer mit der rechten Maustaste auf das Projekt und wählt unter den Android-Tools „Export signed Apk“ aus und folgt den Anweisungen. Für diesen Vorgang benötigt man zur Verifizierung den geheimen Schlüssel des Entwicklers, den ich jedoch nicht in dieser Arbeit veröffentlichen möchte.
Die erstellte apk-Datei kann anschließend über https://play.google.com/apps/publish/ in den PlayStore hochgeladen werden.

3.2.10 Die aufgetretenen Probleme

Während der Arbeit an dem Programm sind kaum Probleme aufgetreten.
Leider ist es mir noch nicht möglich gewesen, die Korrektheit der erfassten Messwerte der Audioanalyse zu verifizieren. Zudem wird die erfolgreiche Anwendung der Audioanalyse bei bestimmten Experimenten durch die Geräusch-Unterdrückung des Android-Betriebssystems verhindert. Sollte über einen längeren Zeitraum dieselbe Frequenz zu hören sein, wird dies durch das Betriebssystem erkannt und herausgefiltert, sodass die Frequenz nicht mehr von der App erfasst werden kann. Leider bietet das System keine Möglichkeit der Deaktivierung dieser Funktion.
Des Weiteren war der Zugang zu einem Mindwave ThinkGear – Headset nicht dauerhaft gewährleistet, wodurch diese Funktion nicht ausgiebig getestet werden konnte.

4 Die Entwicklung des Servers

4.1 Der Datenbankentwurf

Zur Verschaffung eines Überblickes über die benötigten Tabellen, habe ich mir zu aller erst eine Mind-Map erstellt, in der ich alle zu sammelnden Daten und Funktionalitäten zusammengeführt habe.
Anschließend konnte ich daraus einen ersten Entwurf erstellen, welchen ich jedoch während der Umsetzung verbessert habe. Das aktuelle Konzept ist der Anlage 3 zu entnehmen.

Die Datenbank besteht aus acht verschiedenen Tabellen. Eine davon beinhaltet die erfassten Messinformationen, mit den Messwerten, der Zeit sowie eine Fremdschlüssel-Zuordnung zu einem Sensor und einem Gerät.
Daraus resultiert, dass es auch eine Tabelle für die verschiedenen Sensoren geben muss, welche Informationen zu den Achsenbeschreibungen, der Messskalierung, dem Messbereich und Produktionsbeschreibungen enthält.
Des Weiteren existiert eine Tabelle, in der alle Geräte identifiziert werden. Hierzu wird der gehashte Wert der W-LAN-Mac-Adresse genutzt.
Zusätzlich gibt es eine Tabelle für die Accounts. Hierfür kann jeder einen Benutzernamen und Passwort angeben und Maschinen mit diesem verknüpfen. Aus diesem Grund gibt es eine weitere Fremdschlüssel-Tabelle, welche die Verlinkungen von Gerät und Account speichert.
Um ebenfalls die Speicherung von Messübersichten mit den Minima, Maxima und Durchschnittswerten der Messgrößen zu ermöglichen, gibt es eine Tabelle *specialValues* in der Datenreihen aus der *dataTable* über einen Fremdschlüssel markiert werden.

Falls Sensoren mehr als drei Messgrößen aufnehmen, gibt es die Tabelle *expandedDataTable*, in der bei Bedarf zusätzliche Messwerte gespeichert werden können.

Grundsätzlich soll es mit diesem Entwurf möglich sein Daten anonym zu senden, ein Gerät temporär einem Account zuzuordnen und somit den Zugriff eines Lehrers auf die Messergebnisse seiner Schüler zu gewähren.
Des Weiteren sollte es mit diesem Entwurf möglich sein, zukünftige Erweiterungen wie z.B. weitere externe Sensoren oder zusätzliche Funktionalitäten einbauen zu können.

4.2 Das Kommunikationsverfahren

4.2.1 Das grundlegende Prinzip

Für die Verdeutlichung des Ablaufes der Kommunikation habe ich ein Handshake-Diagramm für die Datenübermittlung erstellt, mit dem die übertragenen Informationen visualisiert werden.

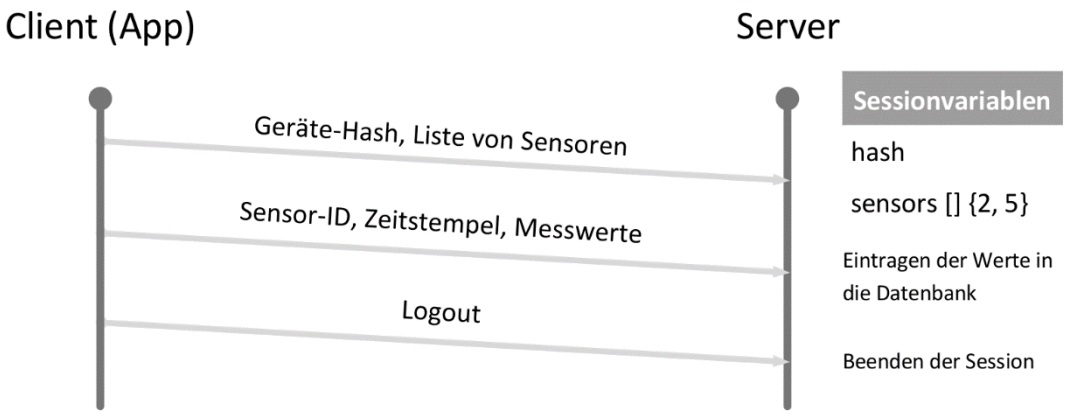


Abbildung 3 Handshake-Diagramm der Messwertübermittlung

Für die Übermittlung der Messwerte ist keine Antwort des Servers auf die Anfrage erforderlich, da standardmäßig der HTML-Response-Code 200 für den Status OK gesendet wird.
Die Sensor-ID des zweiten Schrittes bezieht sich auf den Index des *sensors*-Arrays der Session und ist identisch mit der Reihenfolge der übermittelten Sensoren.
Zusätzlich ist zu erwähnen, dass mehrere Werte in die Datenbank eingetragen werden können, ehe die Session beendet wird.
Ebenfalls ist es möglich zusätzlich zu der Sensor-ID, dem Zeitstempel und den Messwerten eine Variable zu senden, welche den Messwert als Minima, Maxima oder Durchschnitt definiert.

Für die Verknüpfung eines Gerätes mit einem Benutzer-Account habe ich folgendes Handshake-Diagramm erstellt.

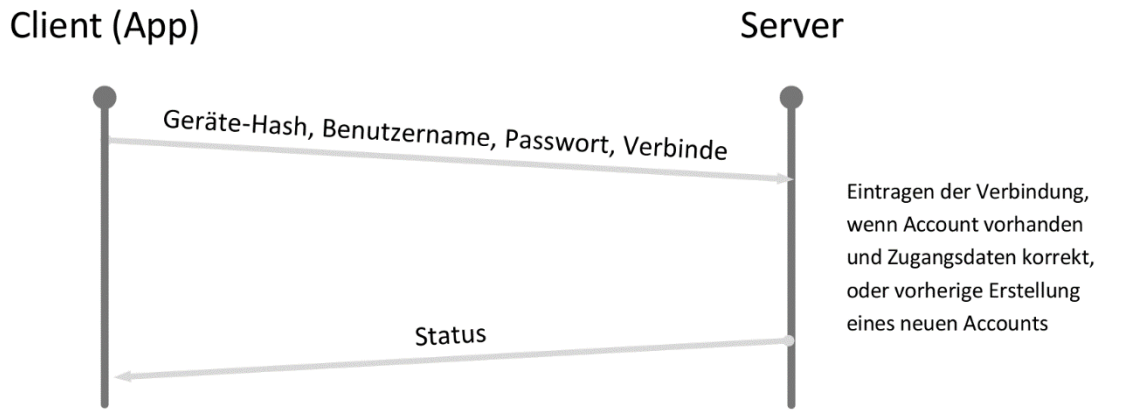


Abbildung 4 Handshake-Diagramm der Accountverknüpfung

Unabhängig von einer laufenden Session ist es möglich, ein Gerät mit einem Account zu verknüpfen. Dazu werden von dem Gerät alle erforderlichen Daten gesendet.
Der Server hat nun wie folgt zu agieren:

Wenn der Account bereits existiert, soll geprüft werden, ob das Passwort korrekt ist. Wenn dies so ist, wird die Verbindung in die *machineUserTable* eingetragen.
Wenn der Account noch nicht vorhanden ist, soll dieser angelegt und mit dem Gerät verknüpft werden.

Ein Benutzer-Account kann immer nur die Daten sehen, welche im Zeitraum von der Verknüpfung und der Überschreibung dieser entstanden sind.
Um die gesendeten Statuscodes richtig interpretieren zu können, soll folgende Tabelle helfen:

Statuscode	Bedeutung
200	Verknüpfung existiert bereits
201	Account und Verknüpfung wurden erstellt
202	Verknüpfung wurde mit bestehendem Account erstellt
401	Passwort ist nicht korrekt (Account existiert bereits)
404	Server auf dem Endgeräte falsch konfiguriert
500	Abfragefehler (serverintern)

Tabelle 1 Bedeutung der Statuscodes des Verknüpfungsvorgangs

4.2.2 Die Entwicklung der PHP-Schnittstelle

Für Auswertung der gesendeten Daten wird eine PHP-Schnittstelle benötigt. Diese kann die POST-Variablen eines Seitenaufrufes auslesen und auswerten.

Grundsätzlich lässt sich die Schnittstelle in zwei Teile untergliedern: der Kommunikation mit dem mobilen Endgerät und der Betrachtung der gespeicherten Daten im Browser.
Beide greifen über die Klasse *connection_manager* auf die Datenbankverbindung zu und können somit Abfragen und Updates übertragen.
Der *datastream_manager* verwaltet die Verbindung mit dem Smartphone indem es eine Session auf- und abbaut, Daten empfängt und Verknüpfungen zwischen Gerät und Account herstellt.
Für die Visualisierung im Web-Front-End ist das PHP-Skript in *menu* zuständig, welches für die Visualisierung in Diagrammform die kostenlose JPGraph[4] – Bibliothek verwendet. Alternativ stand die Bibliothek PCChart[5] zu Auswahl, allerdings gibt es bei der JPGraph-Bibliothek eine Pro-Version, sodass der Support bei dieser gewährleistet zu sein scheint.

In der Weboberfläche soll es möglich sein, sich mit einem Account oder einer Geräte-ID zu identifizieren um anschließend die Messreihen in einem bestimmten Zeitintervall, eines bestimmten Sensors, eines bestimmten Gerätes in einer gewünschten Form, also als Tabelle, Liniendiagramm, Balkendiagramm oder auch CSV-Datei ausgeben zu lassen.

4.2.3 Die Funktionsweise des implementierten Sendevorgangs

Um den gesamten Sendeprozess der implementierten Lösung zu erläutern, habe ich nachfolgend Screenshots von der Postman[6]-Erweiterung des Chrome-Browsers dargestellt. An diesen kann man ein Sendebeispiel nachvollziehen und alle POST-Variablen ablesen.
Postman ist eine Chrome-Erweiterung, welche es einem ermöglich an eine beliebige Webseite POST-Variablen manuell zu übertragen und das Ergebnis vom Server einzusehen.

[4] http://jpgraph.net/download/
[5] http://www.pchart.net/download
[6] https://chrome.google.com/webstore/detail/postman-rest-client/fdmmgilgnpjigdojojpjoooidkmcomcm

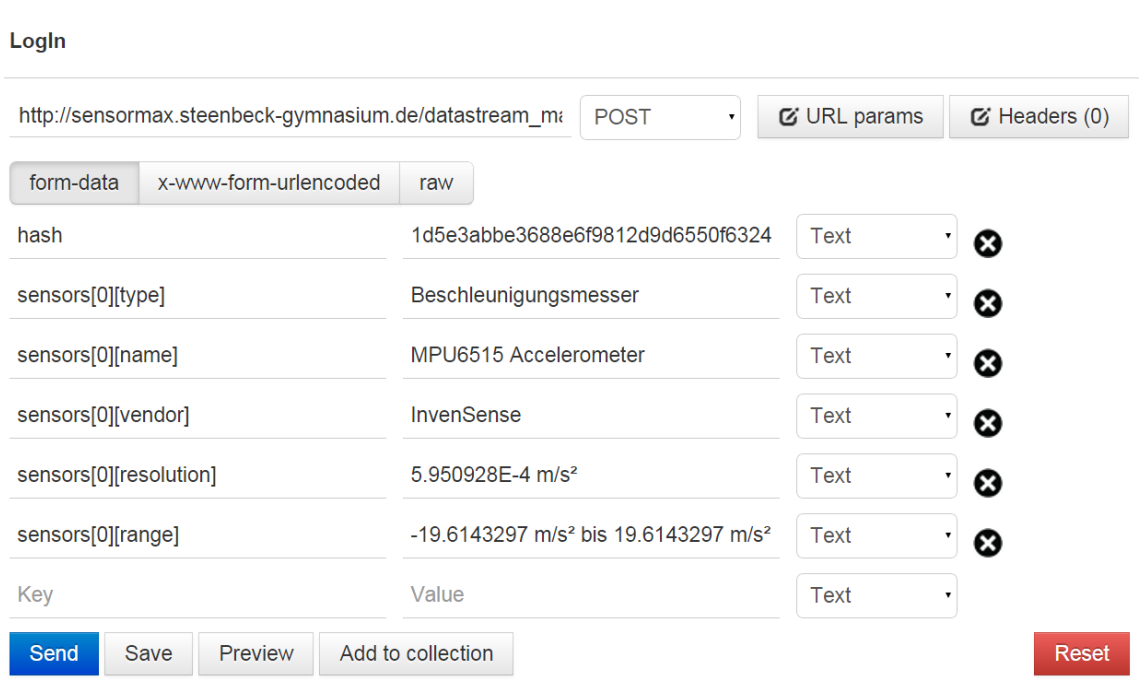


Abbildung 6 Beispiel für einen POST-Request des Login-Verfahrens

Man sendet für den Login zunächst den Hashwert des Gerätes und anschließend ein Array mit den Sensor-Informationen. Wenn man eine Messung mit mehreren Sensoren durchführt, so kann man das Array um *sensors[n][type]*… (n… Nummer des Sensors) ergänzen.

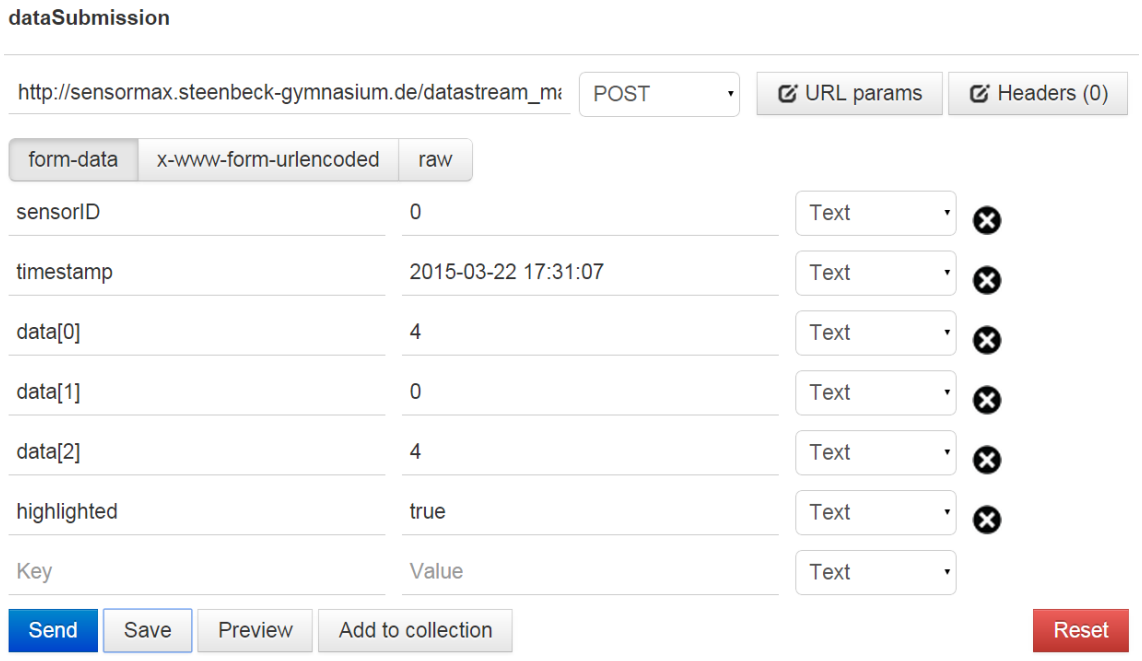


Abbildung 5 Beispiel für einen POST-Rquest der Messwertübertragung

Für die Datenübertragung sendet man den Array-Index für den Sensor. Wenn man also beispielsweise eine Messung mit dem Beleuchtungssensor und dem Beschleunigungssensor durchführt und einen Messwert für den Beschleunigungssensor schickt, so würde der Index 1 betragen, da Index 0 für den Beleuchtungssensor stehen würde. Des Weiteren sendet man die essentiellen Messdaten und eine Variable *highlighted*, welche angibt, ob dieser Messpunkt markiert wurde.

Für den Abmeldevorgang sendet man lediglich die Variable *logoutRequested*. Die Belegung ist dabei nicht entscheidend.

4.2.4 Die aufgetretenen Probleme

Da ich mich in dieser Arbeit mehr oder weniger das erste Mal mit der Scriptsprache PHP beschäftigt habe, traten immer mal wieder unerwartete kleinere Probleme auf. Diese ließen sich jedoch immer mit ein wenig Recherche bzw. ein Blick in die Dokumentation beheben.
Hinderlich waren hingegen die eingeschränkten Rechte auf der Datenbank und die fehlende Verbindung zu dieser, sodass nur über PHP auf diese zugegriffen werden konnte und auch keine Veränderungen in der Struktur möglich waren.

5 Die möglichen Anwendungen

5.1.1 Die Beleuchtungsausmessung eines Sitzplatzes im Hörsaal der BTU

Das Experiment wurde zum letzten Schultag der Jahrgangstufe 11 im Jahr 2014 durchgeführt. Es nahmen ca. 100 Schüler an diesem Experiment teil, indem Sie eine fünfsekündige Messung mit dem Beleuchtungssensor aufnahmen und den Durchschnittswert an einen lokalen AP sandten. Damals verwendeten wir zwar eine Vorgängerversion der App, allerdings bleibt das grundlegende Vorgehen identisch. Im nachfolgenden werde ich Schritt für Schritt die Durchführung beschreiben.

1. Die App öffnen
2. Durch von-links-nach-rechts wischen die Navigation öffnen und die Einstellungen öffnen
3. Den unteren Button anklicken, um zu den Livestream-Einstellungen zu gelangen.
4. Die Serveradresse (http://sensormax.steenbeck-gymnasium.de/datastream_manager.php) eintragen
5. Das Handy mit dem entsprechenden Sitzplatzaccount verknüpfen.
6. Wieder den Slider öffnen und dann die Sensorenübersicht auswählen
7. Den Beleuchtungssensor auswählen und auf öffnen klicken
8. Die gewünschte Messzeit eintragen und den Tooglebutton aktivieren.
9. Messung starten
10. Wenn die Übersicht angezeigt wird, auf *Senden* klicken.

Fall keine Übersicht angezeigt wird, sollte in den Einstellungen geprüft werden, ob der Haken bei *Zeige Messübersicht am Ende* gesetzt wurde.

Eine mögliche serverseitige Visualisierung der Messwerte könnte wie folgt aussehen:

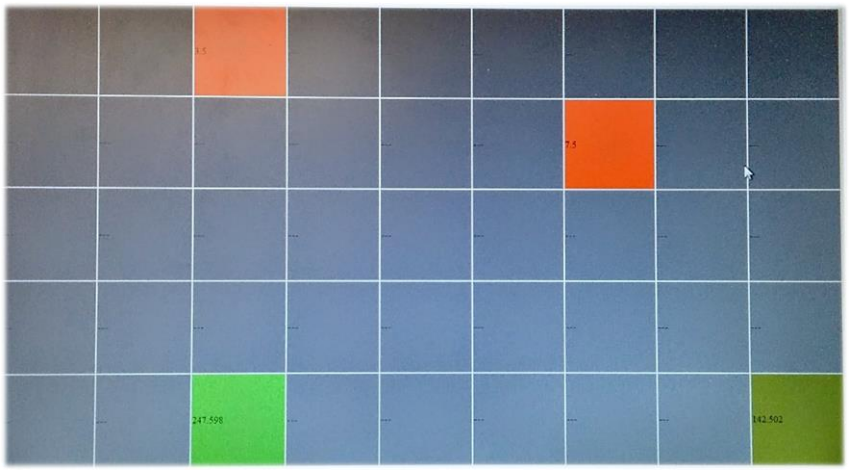


Abbildung 7 Beispielhafte Visualisierung der Messwerte

Der Abbildung kann man nun entnehmen, wie gut ein Raum ausgeleuchtet ist. Die Farben symbolisieren die Helligkeit, rot bedeutet *schlecht ausgeleuchtet* und grün *gut ausgeleuchtet*.

Als Schlussfolgerung könnte man nun Maßnahmen ergreifen, um für jeden Studenten die gleichen Bedingungen zu schaffen.

5.1.2 Experimente mit einzelnen Geräten ohne Erfassung in einem Server

Für den Unterricht ergibt sich eine Vielzahl möglicher Experimente. Besonders im Physikunterricht ist der Einsatzbereich groß. Man könnte z.B. das Magnetfeld um Stromdurchflossene Leiter in Abhängigkeit von Widerstand, Spannung und Stromstärke quantitativ untersuchen. Ebenfalls ist es möglich den Polarisationseffekt des Lichtes nachzuweisen. Für eine derartige Aufgabe wurde in einem Physikwettbewerb von unseren Physiklehrer die Nutzung unserer App empfohlen.

Des Weiteren ist es möglich den Verlauf einer gedämpften Federschwingung aufzunehmen, den Druckunterschied unter einer Vakuumglocke zu verfolgen, oder Experimente mit dem Ton-Generator wie den Dopplereffekt und auch die akustische Schwebung zu untersuchen.

6 Das Feedback

Nach dem Experiment in der BTU erhielt ich von vielen Lehrern und Schülern positive Rückmeldungen. Sie fanden sowohl die Idee gut, also auch die Umsetzung gelungen. Besondere Unterstützung erhalte ich von meinem Informatiklehrer und den Lehrerkollegium des Physikbereiches. Selbst Lehrer anderer Fachbereiche waren begeistert.

Im Jugend-Forscht Wettbewerb erhielt ich ebenfalls viele positive Meinungen, welche mich bekräftigten an dieser App weiterzuentwickeln.

Es unterstützten mich jedoch nicht nur die Lehrerschaft und andere Juroren, sondern auch die Schülerschaft, welche mit konstruktiver Kritik und eigenen Wünschen die Weiterentwicklung der App vorantreibt.

Dies spiegelt sich ebenfalls in der Wertung der PlayStore-Nutzer wieder. Die App wurde seit der Veröffentlichung am 02. Juli 2014 auf 391 Accounts heruntergeladen und ist momentan noch auf 144 installiert. Die App wurde mit 4,93 von 5 möglichen Sternen bei 15 Bewertungen benotet. (MSG-Apps, 2015)

7 DER AUSBLICK

Neben der Implementierung eines Hintergrund-Services habe ich mich bereits mit anderen Erweiterungen beschäftigt. Eine davon ist das sehr interessante Projekt namens OpenTLD. Mit Hilfe dieser weiterentwickelten OpenCV – Bibliothek soll es möglich sein, beliebige Objekte in Echtzeit zu erkennen. Leider funktionierte die Beispiel-Applikation noch nicht sehr gut, weswegen ich beschlossen habe, auf mögliche kommende Verbesserungen zu warten. Einen guten Einblick liefert jedoch dieses Video[7], in dem das Ergebnis jedoch deutlich besser ist, als es bei mir mit derselben App der Fall war.

Ebenfalls sind Erweiterungen und Verbesserungen des Servers und dessen Auswertung möglich.
Um die Unterrichtsgestaltung für Physiklehrer zu vereinfachen, ist es geplant auf einen Blog Experimentieranleitungen zu veröffentlichen.

HINWEIS

Der Quellcode ist über GitHub öffentlich einsehbar:

https://github.com/deeps96/SensorMax

(Es ist ebenfalls möglich, auf eine ältere Version zuzugreifen.)

[7] https://www.youtube.com/watch?v=wTZlGcUOPrY

GLOSSAR

AccessPoint
: Ein lokaler Router. Mit diesem können sich Geräte über W-LAN verbinden und Daten austauschen. 17

Activity
: Eine Klasse, die der Anzeige von visuellen Informationen dient. 8

Array
: Eine Liste festgelegter Dimension. 10

Float
: Datenstruktur für eine Gleitkommazahl 10

Fragments
: Ähnlich einer Activity 8

hashen
: Einen gegebenen Wert mit einem mathematischen Verfahren verschlüsseln. 12

Launcher
: Übersicht aller auf dem Mobiltelefon installierter Applikationen 8

Listener
: Ein Listener ist eine Softwarekomponente, welche man bei einem zweiten Objekt registriert. Dieses Objekt kann nun an den Listener Nachrichten oder andere Daten senden. 9

Manifest-Datei
: Datei welche Angaben zur Version, der verwendeten Activities und den benötigten Berechtigungen enthält. 11

OpenCV
: Eine Bibliothek, welche Algorithmen zur Bildverarbeitung und Objekterkennung enthält. 19

Paket
: Sammlung von Klassen 8

Schlange
: Eine Datenstruktur, bei der das First-In-First-Out - Prinzip gilt. Sie wird durch eine Liste repräsentiert, bei der die zuerst eingefügten Elemente als erstes entnommen werden. 9

while
: Eine Schleife, die solange ausgeführt wird, bis eine bestimmte Bedingung nicht mehr wahr ist. 11

LITERATURVERZEICHNIS

Android Developers. (kein Datum). *Location Strategies | Android-Developers*. Abgerufen am 21. Februar 2015 von Android-Developers: http://developer.android.com/guide/topics/location/strategies.html

Android. (kein Datum). *Sensors Overview | Android-Developers*. Abgerufen am 21. Februar 2015 von Android-Developers: http://developer.android.com/guide/topics/sensors/sensors_overview.html

Google. (kein Datum). *Nexus 5 - Google*. Abgerufen am 20. März 2015 von Google: http://www.google.com/nexus/5/

Kmheide. (29. Januar 2015). *Extensible Markup Language*. Abgerufen am 15. Februar 2015 von Wikipedia, die freie Enzyklopädie: http://de.wikipedia.org/wiki/Extensible_Markup_Language

Madusanka, I. (07. August 2012). *Capturing Sound for Analysis and Visualizing Frequencies in Android*. Abgerufen am 21. Februar 2015 von stackoverflow: http://stackoverflow.com/questions/5511250/capturing-sound-for-analysis-and-visualizing-frequencies-in-android

MSG-Apps. (23. Februar 2015). *Alle Apps - Google Play Developers Console*. Abgerufen am 23. Februar 2015 von Google Play Developers Console: https://play.google.com/apps/publish/?dev_acc=10194552255903733520#AppListPlace

MSG-Apps. (17. Februar 2015). *Sensor-Max - Android Apps auf Google Play*. Abgerufen am 23. Februar 2015 von Google Play: https://play.google.com/store/apps/details?id=com.deeps.sensormax

Ruge, L. (14. Februar 2013). *What does Android's getMaxAmplitude() function for the MediaRecorder actually give me?* Abgerufen am 21. Februar 2015 von Stackoverflow: http://stackoverflow.com/questions/10655703/what-does-androids-getmaxamplitude-function-for-the-mediarecorder-actually-gi

Unbekannt. (13. Februar 2015). *Model View Controller*. Abgerufen am 15. Februar 2015 von Wikipedia, die freie Enzyklopädie: http://de.wikipedia.org/wiki/Model_View_Controller

Unbekannt. (kein Datum). *Android ListView Checkbox Example - OnItemClickListener() and OnClickListener()*. Abgerufen am 21. Februar 2015 von mysamplecode:

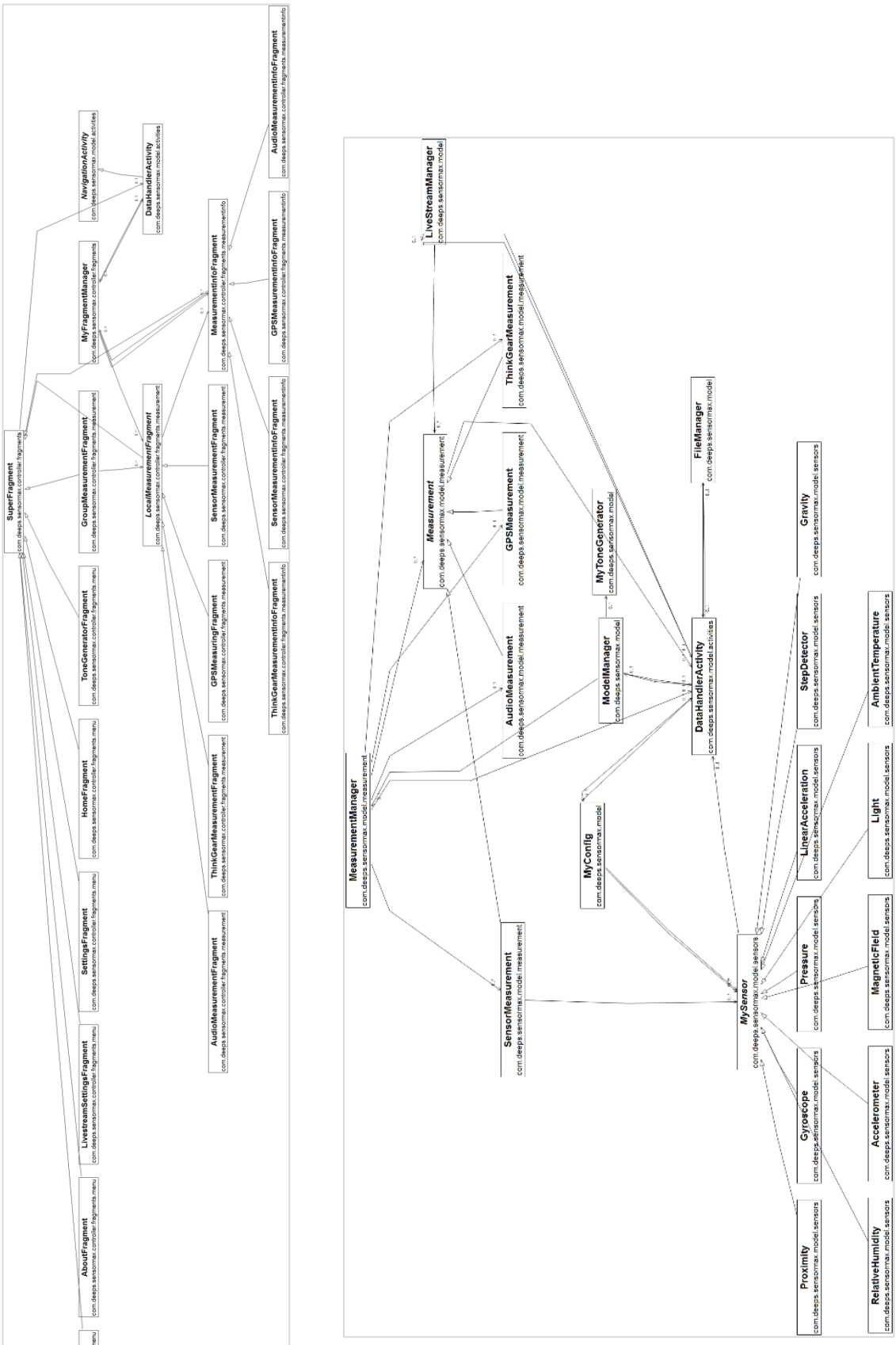
http://www.mysamplecode.com/2012/07/android-listview-checkbox-example.html

Xarph. (26. November 2012). *Playing an arbitrary tone with Android*. Abgerufen am 21. Februar 2015 von Stackoverflow: http://stackoverflow.com/questions/2413426/playing-an-arbitrary-tone-with-android

ANHANG

ANHANGSVERZEICHNIS

ANHANG 1: HIERARCHISCHE ÜBERSICHT ÜBER DIE KLASSENSTRUKTUR



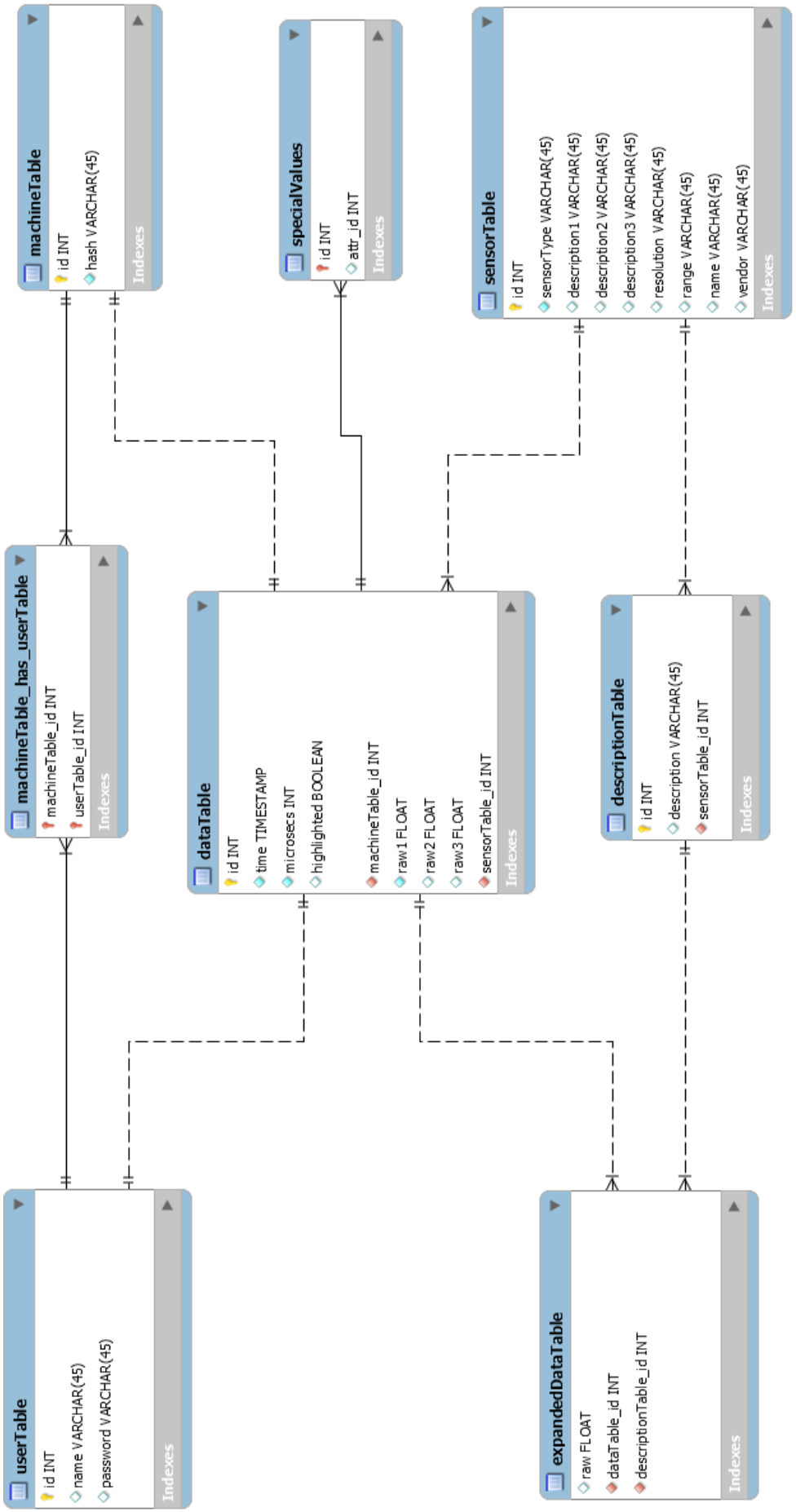
ANHANG 2: QUELLCODEAUSSCHNITT AUS DER MEASUREMENT-KLASSE

```
public void initialise() {
        onStatusChangedListener = new ArrayList<>();
        maxDataCounter = dataHandlerActivity.getMyConfig()
                        .getBufferSizeLocalMeasuring();
        measuringIntervalInMS = dataHandlerActivity.getMyConfig()
                        .getDefaultMeasuringInterval();
        triggerValues = new float[getAxisCount()];
        for (int i = 0; i < triggerValues.length; i++)
                triggerValues[i] = -Float.MAX_VALUE;
        recentDataSet = new float[getAxisCount()];
        zeroing = new float[getAxisCount()];
        min = new float[getAxisCount()];
        max = new float[getAxisCount()];
}

@Override
public void run() {
        isTriggerReleasedFirstTime = true;
        dataCounter = 0;
        data = new float[maxDataCounter][];
        highlightedMeasuringValues = new boolean[maxDataCounter];
        time = new int[maxDataCounter];
        min = new float[getAxisCount()];
        max = new float[getAxisCount()];
        for (int i = 0; i < min.length; i++) {
                min[i] = Float.MAX_VALUE;
        }
        for (int i = 0; i < max.length; i++) {
                max[i] = -Float.MAX_VALUE;
        }
        isTriggerReleased = false;
        registerListener();
        while (state != STATE_STOPPED && !isTimeLimitExceeded()) {
                if (isMemoryExceeded()) {
                        clearLocalBuffer();
                }
                if (state == STATE_RUNNING) {
                        float[] modifiedData = recentDataSet.clone();
                        for (int i = 0; i < modifiedData.length; i++) {
                                modifiedData[i] -= zeroing[i];
                        }
                        if (isTriggerReleased(modifiedData)) {
                                if (isTriggerReleasedFirstTime) {
                                        updateGroupMemberState(state);
                                        isTriggerReleasedFirstTime = false;
                                }
                                calculateMax(modifiedData);
                                calculateMin(modifiedData);
                                if (dataCounter > 0) {
                                        time[dataCounter] = time[dataCounter - 1]
                                                        + measuringIntervalInMS;
                                }
                                updateUIValues(modifiedData, time[dataCounter]);
                                modifiedData = changeDataForStoragePurposes(modifiedData);
                                data[dataCounter] = modifiedData;
                                if (isLiveStreamEnabled) {
                                        sendDataToLiveStream(
                                                        modifiedData,
                                                        System.currentTimeMillis(),
                                                        highlightedMeasuringValues[dataCounter]);
                                }
                                dataCounter++;
                        }
                }
                try {
                        Thread.sleep(measuringIntervalInMS);
                } catch (InterruptedException e) {
                        Log.e(TAG, e.getMessage());
                }
        }
}
```

Quellcode Stand: 26.03.2015

ANHANG 3: DATENBANKENTWURF



EIDESSTATTLICHE SELBSTSTÄNDIGKEITSERKLÄRUNG

Ich versichere, dass ich die vorliegende Seminararbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Texten entnommen sind, wurden unter Angabe der Quellen (einschließlich des World Wide Web und anderer elektronischer Text- und Datensammlungen) und nach den üblichen Regeln des wissenschaftlichen Zitierens nachgewiesen. Dies gilt auch für Zeichnungen, bildliche Darstellungen, Skizzen, Tabellen und dergleichen. Mir ist bewusst, dass wahrheitswidrige Angaben als Täuschungsversuch behandelt werden und dass bei einem Täuschungsverdacht sämtliche Verfahren der Plagiatserkennung angewandt werden können.

Cottbus, 27.03.2015 

Ort, Datum Unterschrift